Can the Eureqa symbolic regression program, computer algebra and numerical analysis help each other?

David R. Stoutemyer*

February 2012

Abstract

The free Eureqa program has recently received extensive press praise. A representative quote is

"There are very clever 'thinking machines' in existence today, such as Watson, the IBM computer that conquered *Jeopardy*! last year. But next to Eureqa, Watson is merely a glorified search engine."

The program is designed to work with noisy experimental data, searching for then returning a set of result expressions that attempt to optimally trade off conciseness with accuracy.

However, if the data is generated from a formula for which there exists more concise equivalent formulas, sometimes some of the candidate Eureqa expressions are one or more of those more concise equivalents expressions. If not, perhaps one or more of the returned Eureqa expressions might be a sufficiently accurate approximation that is more concise than the given formula. Moreover, when there is no known closed form expression, the data points can be generated by numerical methods, enabling Eureqa to find expressions that concisely fit those data points with sufficient accuracy. In contrast to typical regression software, the user does not have to explicitly or implicitly provide a specific expression or class of expressions containing unknown constants for the software to determine.

Is Eureqa useful enough in these regards to provide an additional tool for experimental mathematics, computer algebra users and numerical analysts? Yes, if used carefully. Can computer algebra and numerical methods help Eureqa? Definitely.

Keywords: experimental mathematics, computer algebra, numerical analysis, simplification, symbolic regression, approximation, searching algorithms

*dstout at hawaii dot edu

1 Introduction

Eureqa is a highly praised symbolic regression program described by Schmidt and Lipson [7], freely down-loadable from [9], where there is a bibliography of its use in articles, press citations, a blog and a discussion group. See [6] for quick insight into the underlying Darwinian method for symbolic regression. Eureqa was designed to work from noisy experimental data. However in the first few weeks of using this tool I have already found that also:

1. Eureqa can sometimes do a better job than existing computer algebra systems of exact simplification if an exact result isn't too complicated;
2. Eureqa can often discover simple expressions that *approximate* more complicated expressions or fit a set of numerical results well;
3. Even when the fit isn't as accurate as desired, the *form* of a returned expression often suggests a class of forms to try for focused routine regressions, interpolations or series expansions giving higher accuracy; and
4. Eureqa could do an even better job if it supplemented its search with exploitation of more computer algebra and standard numerical methods, including classic regression;
5. There are important caveats about how to use Eureqa effectively.

This article has examples that illustrate these findings, using Eureqa 0.93.1 beta.

Regarding computing times reported herein, the computer is a 1.60GHz Intel Core 2 Duo U9600 CPU with 3 gigabytes of RAM.

2 Exact simplification and transformation

2.1 A trigonometric simplification example

Here is a Maple 15 assignment of an input trigonometric expression to a dependent variable y:

$$y := \cos(x)^3\sin(x) + \frac{\cos(x)^3\sin(x)}{2} + 2\cos(x)^3\cos(2x)\sin(x) + \frac{\cos(x)^3\cos(4x)\sin(x)}{2} - \frac{3}{2}\cos(x)\sin(x)^3 - 2\cos(x)\cos(2x)\sin(x)^3 - \frac{\cos(x)\cos(4x)\sin(x)^3}{2}. \quad (1)$$

The default simplification merely combines the first two terms, which are similar.

However, then simplify(y) required only 0.03 seconds to return the much simpler

$$4\sin(x)\cos(x)^5\left(2\cos(x)^2 - 1\right). \quad (2)$$

An equivalent expression produced by the *Mathematica* FullSimplify[...] function in only 0.08 seconds is also much simpler:[1]

[1] *Mathematica* output in this article is shown as produced by ending the input with "// TraditionalForm"

$$2\left(\sin(3x) - \sin(x)\right)\cos^5(x)\,. \tag{3}$$

But the equivalent even simpler form discovered by Eureqa is

$$y \;=\; \cos(x)^4 \sin(4x)\,. \tag{4}$$

Caveat: The algorithm uses a random number generator with no user control over its seeding, which is done perhaps by the clock. Therefore sequences are not currently repeatable and the computing times to obtain expression (4) varied dramatically, from 3 seconds to several minutes. It conceivably could require more time than you would ever be willing to invest. However, although even 3 seconds is much longer than the computer algebra times, it is certainly worth a few minutes wait to obtain such a nice result, and such experiments can be done while away from the computer, such as eating, sleeping or playing cell-phone games. There is also an option to use parallel cloud computing, which reduces the mean and variance of the elapsed time necessary to obtain a satisfactory result.

Here is how I used Eureqa to obtain this delightfully simple exact result (4):

1. First I plotted expression (1) in *Mathematica*, revealing that it is antisymmetric and that its fundamental period appeared to be π, with higher frequency components appearing to have a minimum period of $\pi/4$. This was confirmed by TrigReduce[y], which returned the equivalent expression

$$\frac{1}{16}\left(\sin(2x) + 6\sin(4x) + 4\sin(6x) + \sin(8x)\right)\,. \tag{5}$$

2. I decided to use evenly-spaced values of x because expression (1) is periodic, bounded and C^∞ for all real x. I guessed that it might help Eureqa discover and exploit the antisymmetry if I used sample points symmetric about $x = 0$. I also guessed that it might help Eureqa discover the periodicities if I used exactly two fundamental periods. I guessed that using 16 samples within the minimum period $\pi/4$ would be sufficient to resolve it quite well, then I doubled that because Eureqa uses some points for fitting and others for error assessment. This works out to 128 intervals of width $\pi/64$ from $-\pi$ through π. I then created a table of 17-digit x y floating-point pairs and exported it to a comma-separated-values file by entering

$$\text{Export}\left[\text{"trigExample.csv"},\ \text{Table}\left[\text{N}\left[\{x,\ldots\},17\right],\{x,-\pi,\,\pi,\,\frac{\pi}{128}\}\right]\right]$$

where the ellipsis was expression (1).[2]

[2]The reason for requesting 17-digits was that I wanted *Mathematica* to use its adaptive significance arithmetic to give me *results* estimated accurate to 17 digits despite any catastrophic cancellations, and I wanted Eureqa to receive a 17th significant digit to help it round the sequences of input digit characters to the closest representable IEEE double values, which is used by Eureqa. It might appear that I am being compulsive here. However, the Eureqa result could contain floating point constants that are approximations to constant expressions such as $\pi/6$ or $\sqrt{3}/2$. Whenever software contains a floating point number that I suspect might closely approximate such an exact constant, I submit the

3. I then launched Eureqa, opened its spreadsheet-like Enter Data tab, then replaced the default data there with mine by importing file trigExample.csv. In the row labeled var I then entered x in column A and y in column B.

4. The Prepare Data tab then showed superimposed plots of the x and y data values and offered preprocessing options that aren't relevant for this very accurate data.

5. Figure 1 shows the Set Target tab:

   (a) The pane labeled The Target Expression suggests the fitting model $y = f(x)$, which is exactly what I want, so I didn't change it.

   (b) The pane labeled Primary Options has check boxes for the desired Formula building blocks used in composing candidate $f(x)$ expressions. The default checked ones are sufficient for the sort of concise equivalent to expression (1) that I am seeking. Therefore I didn't check any of the other offered functions and operators – not all of which are shown on this screen shot.[3] That dialog pane also shows the corresponding building-block Complexity measures, which can be altered by the user. The complexity measure of an expression is the total of the complexities of its parts.

   (c) The drop-down menu labeled Error metric offers different built-in error measures for Eureqa to try optimizing. A bound is usually more reassuring than the alternatives, so I chose Minimize the worst-case maximum error. The documentation suggests that Maximize the R-squared goodness of fit or Maximize the correlation coefficient is more scale and offset invariant, which are desirable properties too. However, the data is already well scaled with means of 0.

   (d) For the covered Data Splitting drop-down menu the default alternative is to designate a certain percentage of the data points for fitting the data (*training*) and a certain percentage for assessing the error measure (*validation*). The individual assignments to these categories are done randomly, with some overlap if there aren't many data points. For the almost exact data in this article I would prefer that alternate points be assigned to alternate categories, with the

floating point number to the amazing free Internet Inverse Symbolic Calculator [2]. The complexity of the constants that it can identify increases with the precision of the floating point number, and 16 digits of precision is near the minimum necessary to identify linear functions of π, e, and square roots of integers with rational coefficients. Maple contains an earlier version of Inverse Symbolic Calculator named identify(...).

The y values could be generated from the x values *within* the Eureqa Enter Data tab by entering the formula "= ...", where the ellipsis is the right side of formula (1). However, Eureqa rounds such formula-generated values to 15 significant digits. The x and y values can also be generated in Excel, then copied and pasted into Eureqa. However, Excel tends to round values to 10 significant digits. Such rounding might be welcome for aesthetic reasons with low-accuracy experimental data, but it is an unnecessary loss of accuracy for experimental mathematics.

[3] I could have saved search time by un-checking Division, because the floating point coefficients make a denominator unnecessary for this class of expressions. I could also have saved search time and perhaps obtained a more accurate result by checking the Integer Constant box, because integer coefficients are quite likely for exact equivalent expressions, making it worth having Eureqa try rounding to see if that improves the accuracy. However, I decided to accept the default checked boxes to see if Eureqa could find a good exact equivalent without any more help from me. Other building blocks are described at [4].

end points being used for training. However, none of the alternatives offered this so I chose the default.[4]

6. I then pressed the Run button on the Start Search tab and watched the temporal progress of the search. The plot in Figure 2 dynamically zoomed out to show a log-log plot of the lower left envelope of the most accurate candidate obtained so far as a function of computing time. The Project Log dynamically showed successive candidates that are better than any so far on the basis of either complexity or accuracy. The entertainment of watching the evolution of these panes is very appealing. It is similar to rooting for the home team at a sporting event. Notice that:
   (a) Integer powers of sub-expressions are represented in the Project Log as repeated multiplications, and their complexity is measured that way. For example, several results are displayed as
   $$y = \cos(x) * \cos(x) * \cos(x) * \cos(x) * \sin(4*x)$$
   with a complexity measure of 26.[5]
   (b) This result appears first with a reported Fit of 0.00111, followed by the same expression with monotonically better fits up through 3.81×10^{-10} at 39 seconds.[6] I terminated the search at 8 minutes, during which Eureqa tried different formulas that didn't fit as well no matter how complicated they were.
7. Figure 3 shows the results on the View Results tab:
   (a) In the pane labeled Best Solutions of Different Sizes, the first column lists the complexity measure, the second column lists an error measure, and the third column lists the corresponding candidate equation. If you click on a row, the pane in the lower left corner gives more detail, with various rounded error measures including the Primary Objective that I chose, which was maximum absolute error.
   (b) Eureqa does some automatic expression simplification: Whenever a *mutation* produces a new candidate expression or a *crossover* produces two new expressions containing mixtures of the two parents' sub-expressions, a few transformations are applied to make the expression more nearly canonical and to

[4] I have since learned that with such precise data another promising alternative would be to use all of the points for both training and validation, which is a current option.

[5] The complexity measure seems less than ideal – the *ultimately* reported formula in the View Results tab of Figure 3 is $\cos(x)^4 \sin(4x)$. This is more concise and requires only *one* cosine and *two* multiplies to compute the $\cos(x) * \cos(x) * \cos(x) * \cos(x)$ factor if compiled by any decent compiler.

[6] The fact that the same displayed formula was associated with such dramatically different error measures is because Eureqa always round its *displayed* coefficients to about four significant digits, and in this case it enabled display of an exact result that it was merely converging on. If I had checked the Integer Constant building block, then Eureqa probably would have rounded the coefficients to integers. However, $y(0.0) = 0.0$ and the data varies between about ± 3.8, so a maximum residual of 3.81×10^{-10} is enough to convince most people that the *displayed* result is exact. This can be proved by entering

$$\text{TrigReduce}[\ldots - \text{Cos}[x]^4\,\text{Sin}[4x]]$$

with the ellipsis being expression (1). This input returns 0, proving equivalence.

avoid having a misleadingly high complexity on account of uncollected terms, uncombined numeric sub-expressions, etc. This minimal computer algebra is also used to make the displayed results in the Best solutions of different sizes pane more attractive. These transformation include:

- Integer powers and products of sums are expanded to make them more nearly canonical.
- Factors and terms are sorted, then similar factors and terms are collected.
- Numeric sub-expressions are reduced to a single number.
- A few rules such as $\text{abs}(\text{abs}(u)) \to \text{abs}(u)$ and $1 * u \to u$ are applied.

However, you will often see a sub-expression such as $1\cos(1x)$. This is because the 1s are a result of rounding non-integer coefficients to the typically-displayed 3 or 4 significant digits. Nonetheless I noticed definite unexploited opportunities such as Eureqa unnecessarily trying the sub-expression $\sin(x + 6.283)$. Angle reduction and exploitation of odd or even symmetry could preclude the need to try candidates having constant terms outside a much smaller interval. Perhaps rather than gradually implementing more custom computer algebra, Eureqa could embed one of the many free computer algebra systems compared at [13], many of which are conveniently down-loadable from [10].

(c) The pane in the lower right corner plots an error measure as a function of complexity for the 12 reported optimal candidates, with the currently highlighted candidate as a larger red dot.

(d) The pane in the upper right corner shows a plot of the highlighted expression superimposed on the validation points in dark blue and the training points in lighter blue. The drop-down menu above it offers the option of instead plotting the residuals at all of the data points, which gives a much better idea of the fitting errors as a function of x. If these residuals had revealed a non-negligible recognizable pattern such as being approximately of the form $\sin(16x)$, then I would have tried another search with the model $y = f(\cos(x), \sin(4x), \sin(16x))$. I would iterate this process until there was no further improvement or the residuals revealed no non-negligible recognizable pattern.

Instead of searching for the most concise equivalent, one can also search for equivalent expressions of a *particular form*. For example, we could request that expression (1) be transformed into a function without cosines by un-checking that building block. As another example, given an expression $z = \ldots$ that depends on x and y, perhaps it is desired to transform it to an expression depending only on x times an expression depending only on y. This can be requested by making the target expression be $z = f_1(x) * f_2(y)$. (Nonnegative integer suffixes on variable and function names are pretty-printed as subscripts.)

2.2 Some other exact simplification examples

Using "Assuming[. . . , FullSimplify[. . .]]" in *Mathematica* and "simplify(. . .), assuming . . ." in Maple, I could not make them accomplish the following quick successful Eureqa sim-

plifications, for which I selected the default building blocks, Integer Constants, and other functions or operators that occur in the specific given expressions:

$$\left\lfloor \sqrt{\lfloor x \rfloor} \right\rfloor + \left\lfloor \sqrt{x} \right\rfloor \ \mid x \geq 0 \ \ \rightarrow \ \ 2 \left\lfloor \sqrt{x} \right\rfloor . \tag{6}$$

$$q^2 + \frac{2^{1/3}\left(\sqrt{81q^2-12}-9q\right)^{2/3}+24^{1/3}}{6^{2/3}\left(\sqrt{81q^2-12}-9q\right)^{1/3}} - \frac{2\cos\left(\frac{1}{3}\operatorname{acos}\left(-\frac{3\sqrt{3}q}{2}\right)\right)}{\sqrt{3}} \ \mid \ |q| < \frac{2}{3\sqrt{3}} \rightarrow q^2. \tag{7}$$

$$\max(x-y,\,0) - \max(y-x,\,0) \ \ \rightarrow \ \ x - y. \tag{8}$$

- Example (6) comes from [5].
- I tabulated only the *real* part for example (7) because rounding errors for the principal branch of the fractional powers of the negative quantity $\sqrt{81q^2-12}-9q$ generated some relatively very small magnitude imaginary parts.
- To generate $17^2 \rightarrow 289$ rows of data for example (8), I used

$$\text{Apply[Join,Table[N[\{}x, y, \text{Max}[x-\frac{2}{3\sqrt{3}}y, 0]-\text{Max}[y-x, 0]\text{\}}, 17], \{x, -1, 1, \frac{1}{8}\}, \{y, -1, 1, \frac{1}{8}\}].$$

The examples so far illustrate that Eureqa can sometimes determine a simpler exact result than a computer algebra system. However, instead of choosing random examples or ones from the literature, I constructed these very simple results, then obfuscated them in ways that I thought would be difficult for common transformations to reverse – particularly if applied to the entire expression rather than well chosen pieces. Despite this it was not easy to find examples for which neither FullSimplify[...] nor simplify(...) could determine the very simple equivalent. Thus, Eureqa should be regarded as an occasionally beneficial supplement to these functions rather than a replacement. Also, effective use of Eureqa requires good judgment in:

- the interval spanned by the sample points, their number and spacing;
- the form selected for the target expression;
- the Building blocks that are selected together with their complexities, and
- perhaps the error measure that is selected.

Effective use also probably depends on experience with Eureqa. Therefore the benefits that I have discovered should be regarded as a lower bound because of my novice status.

2.3 How to reduce the curse of dimensionality

As illustrated by example (8), data for representing all combinations of n different values for each of m different independent variables requires n^m rows, and we must have $n \geq 2$ for Eureqa to discern any non-constant dependence on each variable. Thus for a given n, this exponential growth in data with the number of independent variables can greatly increases the time required for Eureqa to find a good fit. Yielding to the consequent temptation to save time by reducing n as m increases tends to reduce the precision of the results. Fortunately, there are several complementary techniques that sometimes reduce the effective dimensionality:

1. *Dimensional analysis* can sometimes reduce the number of independent variables, and [11] describes a Maxima program that does this.
2. If you can partition an expanded form of your expression into two or more sums having disjoint variable sets, then you can independently fit each of those sub-sums.
3. If you can partition a factored form of your expression into two or more products having disjoint variable sets, then you can independently fit each of those sub-products.
4. If in every term of a given sum the exponents of some subset of the variables sum to the same homogeneity exponent k, then substitute 1 for one of those variables v, fit this isomorphic problem, then multiply every resulting term by the individual power of v necessary to restore homogeneity exponent k
5. Perhaps some other change of variables such as a matrix rotation can totally or partially decouple the independent variables.
6. Unlike many experimental situations, for experimental mathematics we usually have complete freedom to choose the data values of the independent variables. In both univariate and multivariate problems there can be *sweet spots* that deliver more accuracy for a given number of samples than does a brute force Cartesian product of uniformly spaced points. These special values are often related to the zeros or extrema of orthogonal polynomials. For example, see the the multidimensional integral formulas in Chapter 25 of [1].

3 Approximate symbolic results

Ideally a floating point result is either exact or the closest representable number to the exact result, with ties broken in favor of having the last significand bit be 0. Well designed floating-point *arithmetic* comes very close to this ideal: If the exact result is representable within the thresholds for overflow and for gradual underflow, then the result is the exact result for inputs that differ from the actual inputs by factors between $1 \pm \varepsilon_m$, where *machine epsilon* ε_m is about 1.11×10^{-16} for IEEE double, which corresponds to about 16 significant digits. A relative error bound of $1 \pm \varepsilon_m$ is the *gold standard*. There are similar expectations for operations such as exponentiation and functions such

as sinusoids or logarithms that are commonly built into compilers – but allowing a few times ε_m, which I call the *silver standard*. It is a matter of professional pride among numerical analysts designing *general purpose* mathematical software to strive for the silver standard for all functions provided with the software, such as Bessel functions, etc. I was curious to know if Eureqa can help numerical analysis by discovering concise approximate expressions that meet the silver standard.

3.1 An antiderivative example

Suppose that I want to implement an IEEE double version of the Dogbert W function defined by

$$W(x) \;:=\; \int_0^x \frac{dt}{\sqrt{1 - t^2 + \frac{2}{\pi}\cos\left(\frac{\pi t}{2}\right)}}. \tag{9}$$

No computer algebra systems that I tried can determine a closed form for $W(x)$.

Plan B: Do a gold-standard numeric integration for a sequence of x values in the interval -1$\leq x \leq$ 1, then use Eureqa to discover a silver-standard expression that fits those values well. Here is an abridged account of my attempt to do this.

1. I entered the *Mathematica* input

$$\text{xWPairs} \;=\; \text{Table}\,[\{\text{N}\,[x, 17],\ \text{NIntegrate}\,[\int_0^x \frac{1}{\sqrt{(1-t)(1+t) + \frac{2}{\pi}\cos\left(\frac{\pi t}{2}\right)}},$$
$$\{t, 0, x\}, \text{PrecisionGoal} \to 17,\ \text{WorkingPrecision} \to 25], \{x, -1, 1, 1/64\}. \tag{10}$$

   Then I inspected the 127 number pairs to make sure there were no imaginary parts, infinities or undefined values.[7]

[7]

- The data in the Enter Data tab must be integers or finite real floats. A fraction, a floating point infinity, a nan or a non-real number is not accepted. If an imaginary part of your data is negligible noise, then replace it with 0.0. Otherwise you must separately fit the real and imaginary parts or the absolute value and the arg, which is the Two-Argument Arctangent in Eureqa. If your data has an undefined value due to an indeterminate form, then replace it with the value returned by your computer algebra limit (...) function. If your table has an infinity, then subtract out or divide out the singularity that is causing it. Don't expect a good fit if you simply omit a value that has infinite magnitude or replace it with an exceptionally large magnitude having the correct sign – either of which can dramatically mislead the search.
- I entered $1-t^2$ as $(1-t)(1+t)$ in input (10) and elsewhere to reduce the magnification of rounding errors by catastrophic cancellation near $x = \pm 1$.
- The reason for WorkingPrecision $\to$ 25 is that I wanted to further reduce this catastrophic cancellation.
- A faster but less accurate way to compute a table for a numeric antiderivative is to use a numeric ODE solver, but the time consumed by NIntegrate[...] was negligible compared to the time consumed by Eureqa and the much greater time consumed by me.

2. All the values were real and finite, so I then entered

$$\text{Export}\,[\text{"antiderivative.csv"},\ \text{xyPairs}]\,,$$

then I imported the resulting file into Eureqa and tried to fit $W = f(x)$ with the default formula building blocks. After three minutes of search the most accurate formula was

$$W \;=\; \frac{0.7773112536\,x}{\cos(0.4696613973\,x^7) - 0.2294845216\,x^2},$$

which is non-insightful and accurate to a disappointing maximum absolute error of about 0.4%.

3. A *Mathematica* plot of the integrand reveals a probable explanation: The integrand has endpoint singularities, and although they are integrable, the resulting integral has infinite magnitude low-order derivatives at the endpoints, which probably makes the quadrature less accurate and the Eureqa search slower than otherwise.

4. The plot of numeric antiderivative values on the Prepare Data tab revealed that they look approximately proportional to $\text{asin}(x)$.

5. So next I did another Eureqa search after disabling the sine and cosine building blocks and making the target expression be $W = f(x, \text{asin}(x))$. I did *not* enable the arcsine building block because, for example, I didn't want Eureqa to waste time trying sub-expressions such as $\text{asin}(1 + 3x^2)$. In 11 seconds Eureqa found the following seven term expression having a maximum absolute error of only about 3.1×10^{-9}:

$$\begin{aligned} W \;=\; & -1.870027576 \times 10^{-13} + 0.7816747744\,x + 0.0147770774\,x^3 - \\ & 0.03033616234\,x^2\text{asin}(x) + 0.07586494202\,x\,\text{asin}(x)^2 + \\ & 0.0818165982\,\text{asin}(x)^3 + 0.0009144579166\,x^3\text{asin}(x)^2. \end{aligned} \tag{11}$$

I aborted the search at 3 minutes with no further improvement. Regarding this result:

   (a) To view the coefficients rounded to 10 significant digits rather than about 4, I had to copy the expression from the Best solutions of different sizes pane in the View Results tab into *Mathematica* or a text editor. Unfortunately there is no way to view all 16 digits, so I will have to polish a result with other software to obtain a silver standard result.[8]

   (b) I edited the copied formula because it contained annoying superfluous parentheses and represented integer powers by repeated multiplication.

[8] The rounding of coefficients in the Project Log and Best solutions of different sizes panes *does* increase comprehensibility – particularly since those panes are regrettably un-scrollable. Even more rounding could justifiably be done for coefficients of terms whose relative maximum contribution is small. For example, if the maximum contribution of a term over all data points is 1%, then its coefficient justifiably could be rounded to 2 less significant digits than the coefficient that contributes the most over all data points. See [3] for more about this idea.

(c) The even symmetry of the integrand and the centered integration from 0 imply that the antiderivative should have odd symmetry. Therefore the spurious constant term in expression (11) should be discarded. Terms that are contrary to odd symmetry might arise from the random partitioning of the data into training and validation sets.

(d) Converting the result to recursive form by collecting similar powers of x or $\text{asin}(x)$ can significantly reduce the Complexity. For example, rounding coefficients the coefficients in result (11) for brevity,

$$W = 0.78\,x + 0.015\,x^3 - 0.03\,x^2\text{asin}(x) + (0.076\,x + 0.00091\,x^3)\,\text{asin}(x)^2 + 0.082\,\text{asin}(x)^3.$$

which saves one instance of "$*\text{asin}(x)^2$. At the expense of legibility, the Complexity and the floating-point substitution time can be further reduced by *Hornerizing* this recursive representation to

$$W = x(0.78 + 0.015x^2) + \text{asin}(x)(-0.03\,x^2 + (x\,(0.076 + 0.00091\,x^2) + 0.082\,\text{asin}(x))\,\text{asin}(x))).$$

This is another place where more computer algebra could help Eureqa.

(e) Notice that result (11) has no term that is simply a numeric multiple of $\text{asin}(x)$. Therefore it doesn't model the infinite endpoint slopes associated with the integrand singularities there. But that is my fault because the target expression contains no special encouragement to include a term of that form, and the plot from the Eureqa Prepare Data tab reveals that the sample points were not closely enough spaced near the endpoints to suggest the infinite slope magnitudes there.

6. Result (11) suggests that a set of particularly good fits would be truncated series of the form

$$(c_1\,\text{asin}\,(x) + c_2\,x) + \left(c_3\,\text{asin}\,(x)^3 + c_4\,x\,\text{asin}\,(x)^2 + c_5\,x^2\text{asin}\,(x) + c_6\,x^3\right) + \cdots,$$

which can be regarded as a truncated bi-variate series in terms having a power of x times a power of $\arcsin(x)$ that have non-decreasing odd total degrees, with numeric coefficients c_k. I could constrain Eureqa to search only within this class of expressions by entering a target expression having a particular total degree, such as

$$W = f_1()\,\text{asin}\,(x) + f_2()\,x + f_3()\,\text{asin}\,(x)^3 + f_4()\,x\,\text{asin}\,(x)^2 + f_5()\,x^2\text{asin}\,(x) + f_6()\,x^3.$$

However, standard regression software is much faster for merely optimizing some parameters in a specific form – and more accurate if done with arbitrary-precision arithmetic.[9] The largest total degree in Eureqa result (11) is 5. Consequently I next

[9]This is another area where other kinds of mathematical software could help Eureqa. Eureqa would probably be much faster if it used standard linear or nonlinear regression while it is merely *adjusting* coefficients in a particular form. Now that Eureqa has demonstrated how well genetic search can do unassisted, there is no reason not to make it much faster with assistance from other disciplines.

used the *Mathematica* LinearModelFit [...] function with the 12 basis expressions of the form $x^j \text{asin}(x)^k$ having odd total degree $1 \le j + k \le 5$. After 0.27 seconds I received the following eight term result whose significant digits I have truncated for brevity:

$$22.9\,x + 0.012\,x^3 + 0.000097\,x^5 - 22.1\,\text{asin}(x) - 0.068\,x^2\text{asin}(x) + \\ 0.78\,x\,\text{asin}(x)^2 + 3.09\,\text{asin}(x)^3 - 0.078\,\text{asin}(x)^5. \qquad (12)$$

The maximum absolute error at the tabulated points was about 4×10^{-11}, which is two more significant digits than (11) for the same total degree and only one more term. Very gratifying.

7. Encouraged, I next tried LinearModelFit [...] with total degree 7, which entails 20 basis terms. After 0.52 seconds I received a different eight term result

$$7.48\,x + 0.0041\,x^3 - 6.7\,\text{asin}(x) - 0.019\,x^2\text{asin}(x) + 0.26\,x\text{asin}(x)^2 - \\ 0.000012\,x^5\text{asin}(x)^2 + 1.01\,\text{asin}(x)^3 - 0.025\,\text{asin}(x)^5 \qquad (13)$$

having an absolute error at the tabulated points of about 1.5×10^{-11}. Thus it seems likely that something important is missing from the basis, preventing results (12) and (13) from corresponding to economized truncations of a convergent infinite series because:

   (a) This is a negligible accuracy improvement.
   (b) The term count didn't increase.
   (c) The coefficients of similar terms change dramatically between approximations (12) and (13).
   (d) In each approximation there is substantial cancelation between the dominant terms.

Thus it is not promising to try larger total degrees with this basis in pursuit of a concise numerically stable silver standard. Nonetheless, this has been a nice combined use of Eureqa and *Mathematica*: After the decision to try a target expression of the form $f(x, \text{asin}(x))$, Eureqa discovered a concise form having 9 significant digits and basis functions of the form $x^j\text{asin}(x)$. I then used *Mathematica* to obtain a result having only one more term that is accurate to 11 significant digits, which is more than adequate for many purposes. For this example Eureqa has served as a muse rather than an oracle.

However, I couldn't help wondering: *Is* there a relatively simple class of forms that Eureqa might have revealed for further polishing to a concise *silver-standard* fit by *Mathematica*?

With the help of the *Mathematica* Series [...] function and an embarrassingly large amount of my time adapting its results to my desires via a sixteen line *Mathematica* function that I wrote, I was able to determine an exactly integrable truncated infinite series expansion of the integrand that converged sufficiently fast without undue catastrophic cancelation over the entire interval $-1 \le x \le 1$. Using Assuming[$-1 \le x \le 1$, Integrate[...]], the Hornerized representation of the corresponding *antiderivative* is

$$\tilde{W}(x) \;:=\; c_0\,\text{asin}(x) + x\sqrt{(1-x)(1+x)}\left(c_1 + x^2\left(c_2 + x^2(c_3 + \cdots)\right)\right). \qquad (14)$$

Being a single series bi-focally expanded about $x = \pm 1$ rather than a conditional expression containing two or more different series, it is C^∞ over the entire interior interval $-1 < x < 1$. Notice that contrary to the previous efforts, $\text{asin}(x)$ occurs only to the first power and that all the other terms are multiplied by $\sqrt{(1-x)(1+x)}$. A good set of basis expressions is thus $\{\text{asin}(x),\, x^{1+2k}\sqrt{(1-x)(1+x)}\}$ for $k = 0, 1, \ldots$. The higher powers of $\text{asin}\,(x)$ in the previous basis were being used as flawed surrogates for $x^{1+2k}\sqrt{(1-x)(1+x)}$: Octagonal pegs in circumscribed round holes – they sort of fit, but in an impaired way .

The *Mathematica* function that I wrote computes the exact coefficients c_k, which involve $\sqrt{6}$ and powers of π. However, I instead used LinearModelFit [. . .] to effectively economize a higher degree approximation to a lower-degree one having nearly the same accuracy. For the data I used high precision numerical integration after using truncated series (14) to subtract out and exactly integrate two terms of the endpoint singularities. This made equally spaced x values quite acceptable. Experiments revealed that using terms through coefficient c_7 gave an approximation to $y(x)$ that had a worst absolute error of about 6×10^{-16} at the right endpoint where the value was about 1.255. A plot of the residuals revealed that this eight-coefficient approximation has a relative error of about 5×10^{-16}, making it silver standard. Moreover, this plot was essentially noise, revealing no remaining exploitable residual for IEEE double.

Eureqa did not by itself discover this better basis, but that is my fault: I never allowed square roots as a Building block. I should have done a little analysis earlier to wisely suggest a target expression of the form $f(x,\, \text{asin}(x),\, \sqrt{(1-x)(1+x)})$. After all, there is a square root in the integrand denominator, so I should have expected a square root in a good approximate antiderivative numerator. I could have even tried the square root that also contains the cosine term. Using Eureqa effectively for computer algebra and numerical analysis should be viewed as a collaboration rather than a competition. And Eureqa is not a black box oracle. You cannot leave your brain behind. That is probably equally important when using Eureqa for noisy experimental data.

3.2 Other possible approximate symbolic results

There are many other possibilities for using Eureqa to find a concise expression or suggest a good class of expressions that closely approximates a result that would otherwise have to be presented only graphically or as a table. For example,

- *Inverse functions and solving parametric algebraic equations*: Imagine that for the example of the previous subsection, we also or instead wanted an approximate symbolic expression for the *inverse* Dogbert W function: x as a function of W. Eureqa can search for such expressions if we simply enter $x = f(W)$, or perhaps more specifically for this example $x = \sin(f_2()\, W) + f_3(W)$.
- *Solution of differential, integral, delay and other kinds of functional equations*: For example, suppose that we have a system of first-order ordinary differential equations, and a tabulated numerical solution vector $[y_1(t), y_2(t), \ldots]$ for $t = t_0, t_1, \ldots t_n$. Then we can use Eureqa to separately search for good expressions for $y_1(t), y_2(t), \ldots$. If we want $t_n = \infty$, then we should first make a change of independent variable to

map that end point to a finite value in a way that doesn't introduce a singularity elsewhere.

- *Implicitization*: A major application of computer algebra is implicitization of parametric equations defining curves, surfaces or higher-dimensional manifolds. However, exact methods are known only for certain classes of parametric equations. For other classes we can try using Eureqa to find approximate implicit equations. (Hint: see [8] for tips on how to use Eureqa for this purpose.)

4 Additional tips and tricks

Here are a few additional tips and tricks for using Eureqa effectively in conjunction with computer algebra or numerical analysis:

1. Building blocks such as asin(...), acos(...), atanh(...), log(...), and pow(...,...) that are not real for some real arguments cause a candidate to be rejected if the result isn't real for *all* of the data values. To avoid wasting time partially processing such candidates, consider using a specific form of such a function in your target expression that automatically precludes unreal sub-expressions rather than simply including the function as a building block. For example, knowing that $-1 \le x \le 1$, I used $W = f(x, \mathrm{asin}(x))$ for the example in subsection 3.1 rather than merely checking the Arcsine building block.
2. Conversely, your target expression can be of the form "$\ldots = f(x)+0*\log(f(x)-7)$" to impose an inequality constraint of the form $f(x) > 7$. Whenever $f(x) \le 7$ the log factor generates a nan, and 0*nan is nan, which disqualifies that candidate. Otherwise the multiplication by 0 makes that term have no effect on the search for good $f(x)$, and we can mentally discard the artifice log term at the end. Similarly a target expression of the form "$\ldots = f(x) + 0 * \mathrm{asin}\left((f(x) - 7)/3\right)$" imposes the inequality constraint $4 \le f(x) \le 10$. (Perhaps a future version will have a more efficient and straightforward way to impose inequality constraints.)
3. To increase the chance of obtaining exact rational coefficients, reduce your input expression over a common denominator, then separately tabulate and fit the numerator and denominator with the Integer Constant building block checked.
4. To possibly include *foreseen* exact symbolic known irrational constants such as π and $\sqrt{2}$ in your result, for each such constant make a labeled column of corresponding floating-point values, preferably accurate to 16 digits, For example if a column labeled pi contains 3.14159... and a column labeled $sqrt2$ contains 1.41421..., then check the Integer Constant building block and use a target expression of the form "$\ldots = f(pi, sqrt2, \ldots)$". These symbolic constants don't contribute much to the curse of dimensionality because they are invariant, so there is no need to include more rows on their account.
5. As with most regression software, Eureqa tends to express results in terms of the notoriously ill-conditioned monomial basis $1, x, x^2, x^3, \ldots$ To express your result more accurately in terms of an orthogonal polynomial basis, such as the Chebychev T

polynomials, generate columns of the desired number of successive polynomials at the tabulated values of the independent variables labeled $T_0, T_1, \ldots$, then use a target expression of the form "$\ldots = f(T_0, T_1, \ldots)$". These numerous T_k variables don't contribute much to the curse of dimensionality because they are totally correlated rather than truly independent. Moreover, these variables have the same Complexity as any other variable, thus preventing them from being discriminated against as they would be if you otherwise equivalently made the target expression be "$\ldots = f(1, x, x^2 - 1, x^3 - 3x, \ldots)$".

6. Eureqa has a built-in function $D(columnVariable_1, columnVariable_2, n)$ that uses numerical differentiation of a smoothing spline to approximate an nth derivative. One of its most common uses is for system identification: Given experimental tabulated t y pairs, if you enter a target expression such as $D(y, t, 2) = f(y, D(y, t, 1), t)$, then Eureqa will search for a concise differential equation of that rather general form that accurately fits the data. For example, the returned results might include $D(y, t, 2) = x^2 y^2 D(y, t, 1) + y^3 + \cos(t)$. Although differentiation of a smoothing spline is probably significantly more accurate than more naive methods – particularly for noisy data – numerical differentiation inherently magnifies rounding and discretization errors. Therefore if an explicit or implicit expression is available for y, then it is almost certainly more accurate to tabulate floating-point values of exact derivatives produced by computer algebra. Automatic differentiation [14] is another silver-standard alternative.

7. Eureqa has a built-in integral(...) function that I could have used it instead of the *Mathematica* NIntegrate[...] function. However, the integral(...) function uses the trapezoidal rule – probably because it has no control over the spacing of the independent variable values. Given an integrand expression to integrate exactly or to sample adaptively as needed by a function such as NIntegrate(...), computer algebra can almost always deliver a more accurate integral.

5 The need for seamless integration of separate software packages

It is usually somewhat of a nuisance to work between two or more separate software packages compared to working entirely within one that has all of the necessary features built in. The extra effort and the lesser chance of even knowing about independent software packages is a great deterrent to such use by the masses. Thus it is good to know that interfaces are under development for using Eureqa from within *Mathematica*, Matlab, Python, .NET and KNIME, with current versions down-loadable from [9]. Such added interfaces are rarely as seamless as features that are built in, but they are often great improvements over communicating via export-import of files or copy and paste. Among other things, the interfaces can accommodate differences in syntax for expressions.

6 The need for automation

At any one time, most of us are amateurs with most of the software that we use. We need all of the help that we can obtain. In response to that need, many of the best software packages are automating portions that can be. For example:

1. Adaptive quadrature relieves us of having to learn about numerous rules for different kinds of univariate and multivariate integrands and finite, semi-infinite or infinite integration regions.
2. Adaptive interval and significance arithmetic can give us results that are guaranteed or highly likely to have the accuracy that we request.
3. Eureqa can sometimes automatically discover good regression models.
4. Inverse Symbolic Calculator can sometimes automatically recognize floating-point approximations to exact symbolic rational or irrational constants.

However, the examples in subsections 2.1 and 3.1 were *not* automatic. Training, experience and judgment were involved in choosing the data points, target expression, etc. Some of this could and should be automated. A qualitative analysis program such as the Maxima program described in [12] could help in these regards: Given an expression, this program attempts to return bounds and indications of monotonicity, convexity, symmetries, periodicities, zeros, singularities, limits and stationary points. To the extent that this is successful, this information could be used to choose automatically for each independent variable the endpoints, the number of samples, and perhaps even their distribution. For example:

- If there are singularities, then it is best to automatically convert any tangents to sines divided by cosines, then form a reduced common denominator, then use Eureqa to fit separately the numerator and denominator. Neither the numerator nor denominator will contain a pole, but they still could contain a logarithmic singularity that would have to be handled by subtracting or dividing it out.
- It is probably best if the endpoints extend modestly beyond all of the real zeros and stationary points, including at least one fundamental period if any, with enough points to resolve the shortest period.
- If there is a symmetry and the expression is C^∞ at the symmetric point, then it is probably best to center the data values on that point. Otherwise it might be more efficient to make that symmetry point be one of the end points.

Even a purely *numeric* program that searched for zeros, singularities, extrema, periodicities, and symmetries could help in these regards.

7 Conclusions

1. With wise use, Eureqa can sometimes determine a simpler exact equivalent expression than current computer algebra systems can – or sometimes transform an expression into a targeted form that isn't provided by a computer algebra system.

2. With wise use, Eureqa can sometimes suggest promising forms of expressions that approximate a result for which an exact closed form is unobtainable or excessively complicated. Often you will want to obtain a more accurate result in a thus-revealed class by using a linear or non-linear regression program built into a computer algebra system or statistics package.
3. Some simple interface additions *within* Eureqa could increase its utility for the above two purposes. Interfaces *to* Eureqa within more other software packages would encourage more use of Eureqa. Embedding Eureqa within those packages would probably be even more seamless.
4. Eureqa uses some custom computer algebra and perhaps some standard numerical methods internally. Eureqa would almost certainly benefit from embedding and exploiting a well developed full-fledged computer algebra system together with classic regression and numerical methods packages.
5. Eureqa's most frequent and notable successes will probably continue to be with noisy experimental data, but Eureqa shows promise for purposes 1 and 2 above.

Acknowledgment

Thank you Michael Schmidt for your suggestions and patient explanations.

References

[1] M. Abramowitz and I.A. Segun, Handbook of Mathematical Functions, Dover Publications, (1965).

[2] D. Bailey and J. Borwein, Inverse symbolic calculator, `http://isc.carma.newcastle.edu.au/`

[3] R. Corless, E. Postma and D. R. Stoutemyer, Rounding coefficients and artificially underflowing terms in non-numeric expressions, *ACM Communications in Computer Algebra* 45 (1/2), (2011), pp. 17-48.

[4] Eureqa building blocks, `http://creativemachines.cornell.edu/eureqa_ops`

[5] R. L. Graham, D. E. Knuth and O. Patashnik, Concrete Mathematics, 2nd edition, Addison Wesley, Section 3.2, (1994).

[6] J. R. Koza, Example of a run of genetic programming, `http://www.genetic-programming.com/gpquadraticexample.html`

[7] M. Schmidt and H. Lipson, Distilling free-form natural laws from experimental data, *Science* 324 (5923) (2009), pp. 81-85.

[8] M. Schmidt and H. Lipson, Symbolic regression of implicit equations, Genetic Programming Theory and Practice, Volume 7, Chapter 5, pp. 73-85, (2009), `http://creativemachines.cornell.edu/sites/default/files/Schmidt-Final-2009-06-05.pdf`

[9] M. Schmidt, Eureqa web page. http://creativemachines.cornell.edu/eureqa

[10] SourceForge, http://maxima.sourceforge.net/compalg.html

[11] D. R. Stoutemyer, Dimensional analysis using computer symbolic mathematics," *Journal of Computational Physics* 24, No. 2, (1977), pp. 141-149.

[12] D. R. Stoutemyer, Qualitative analysis of mathematical expressions using computer algebra, Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation, (1976) pp. 97-104.

[13] Wikipedia, Comparison of computer algebra systems, http://en.wikipedia.org/wiki/Comparison_of_computer_algebra_systems

[14] Wikipedia, Automatic differentiation, http://en.wikipedia.org/wiki/Automatic_differentiation

Figures

Figure 1: Eureqa Set Target tab for exact trigonometric simplification example

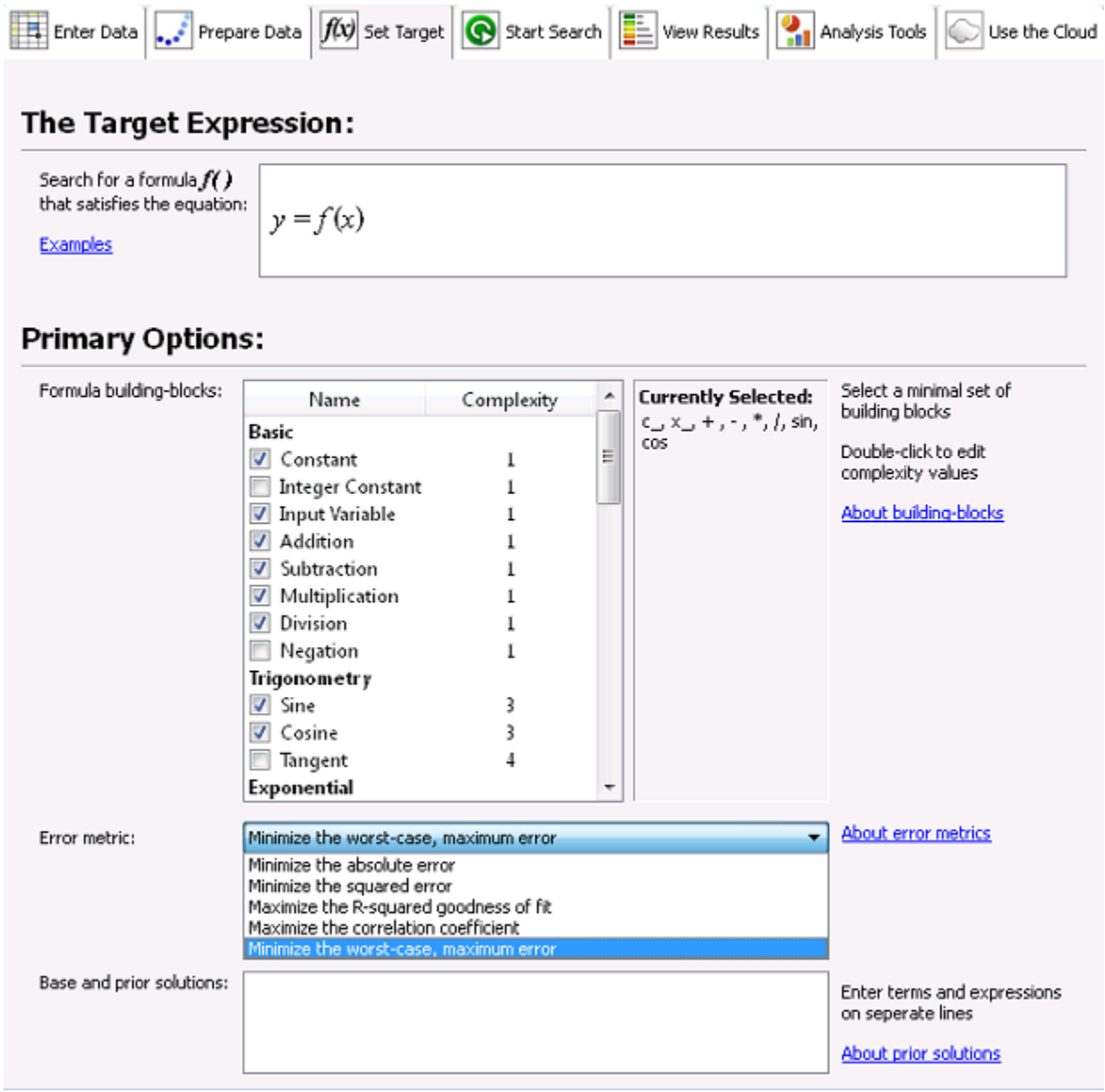

Figure 2: Eureqa Start Search tab for exact trigonometric simplification example

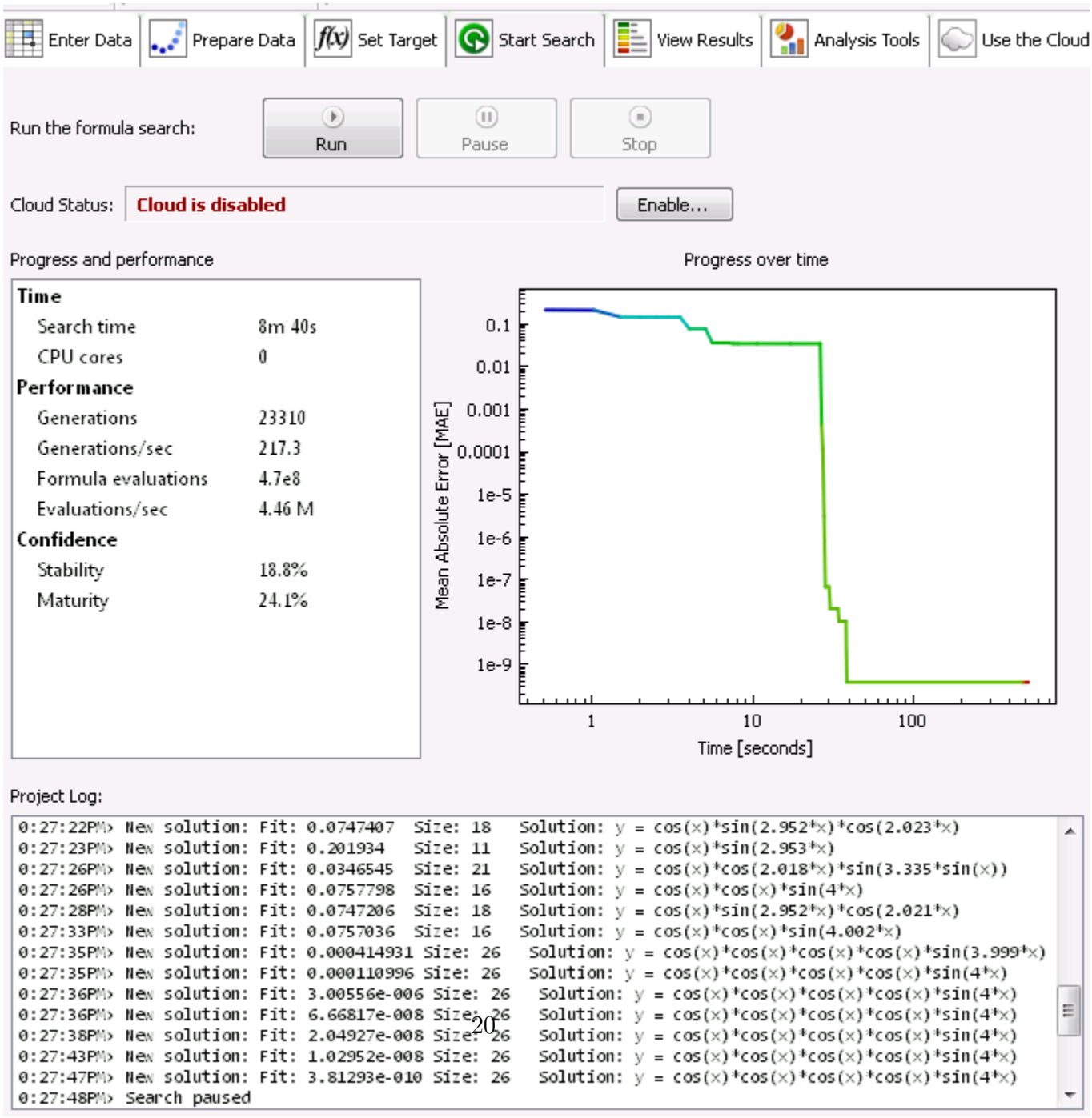

Figure 3: Eureqa View Results tab for exact trigonometric simplification example